Logical Agents. Logic. Propositional Logic

Outline

- Knowledge-based agents
- Wumpus world
- Logic in general - models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving

Knowledge bases

Knowledge base = set of sentences in a formal language
Declarative approach to building an agent (or other system):

- `Tell` it what it needs to know

Then it can `Ask` itself what to do - answers should follow from the KB

Agents can be viewed at the knowledge level
i.e., what they know, regardless of how implemented

Or at the implementation level

- i.e., data structures in KB and algorithms that manipulate them

A simple knowledge-based agent

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to:

- Represent states, actions, etc.
- Incorporate new percepts
- Update internal representations of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

PEAS description

Performance measure

Environment

Actuators

Sensors

For example, for an Interactive English Tutor system:

- Performance measure – student score on test
- Environment – set of students, tests
- Actuators – display exercises, corrections, feedback
- Sensors – keyboard input

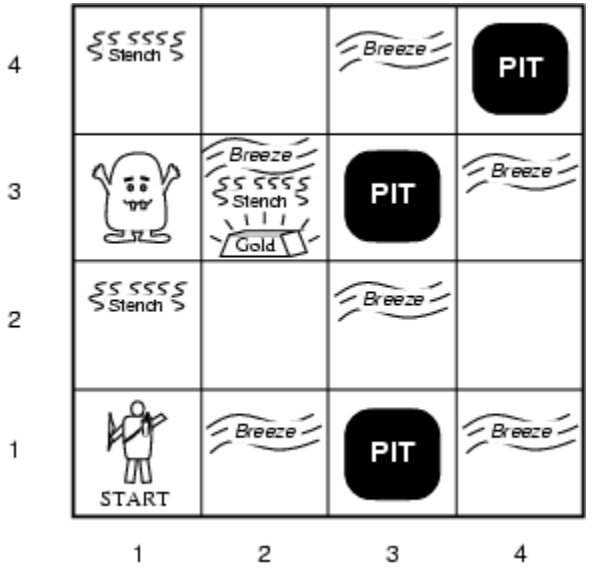
Wumpus World PEAS description

Performance measure:

- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment:

- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square



Sensors: Stench, Breeze, Glitter, Bump, Scream

Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus world characterization

Fully Observable? No – only local perception

Deterministic? Yes – outcomes exactly specified

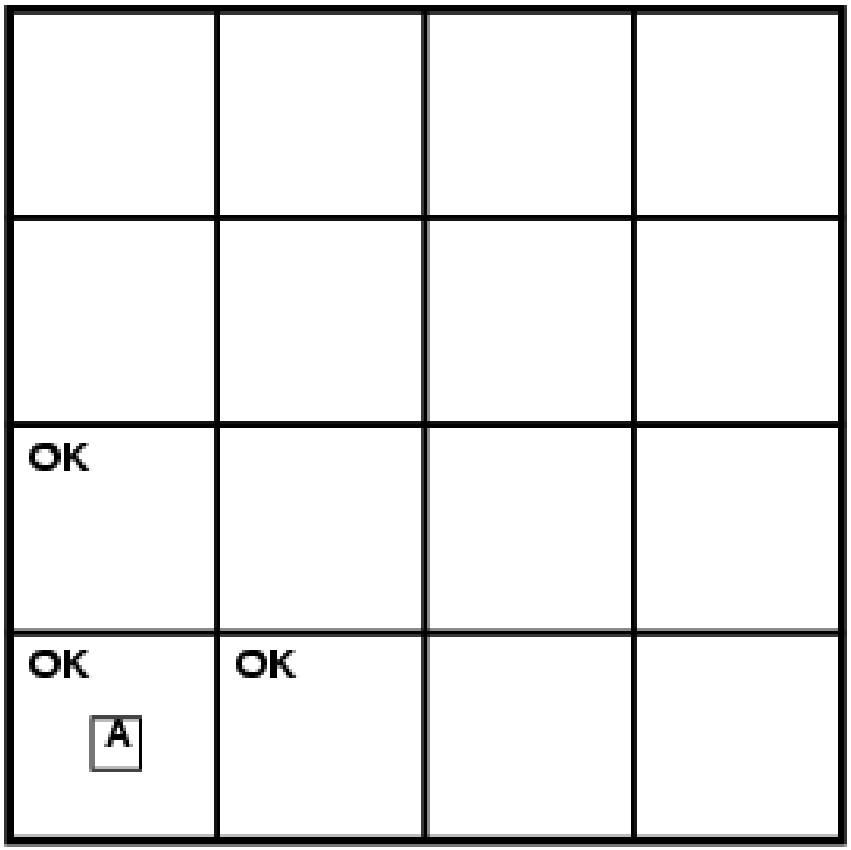
Episodic? No – sequential at the level of actions

Static? Yes – Wumpus and Pits do not move

Discrete? Yes

Single-agent? Yes –Wumpus is essentially a natural feature

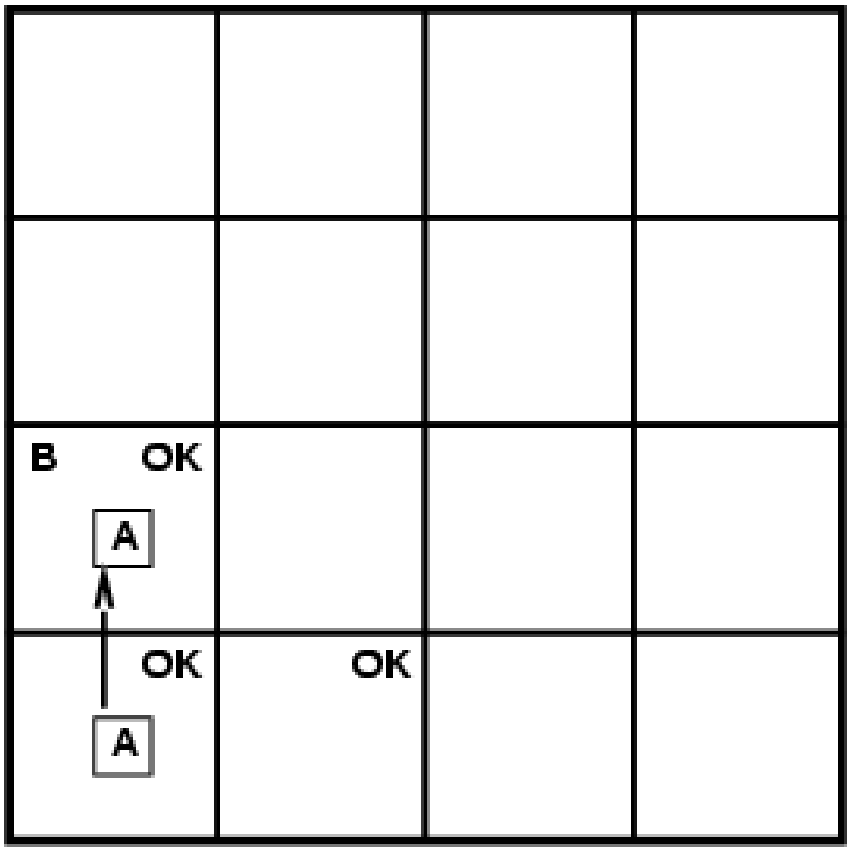
Exploring a wumpus world



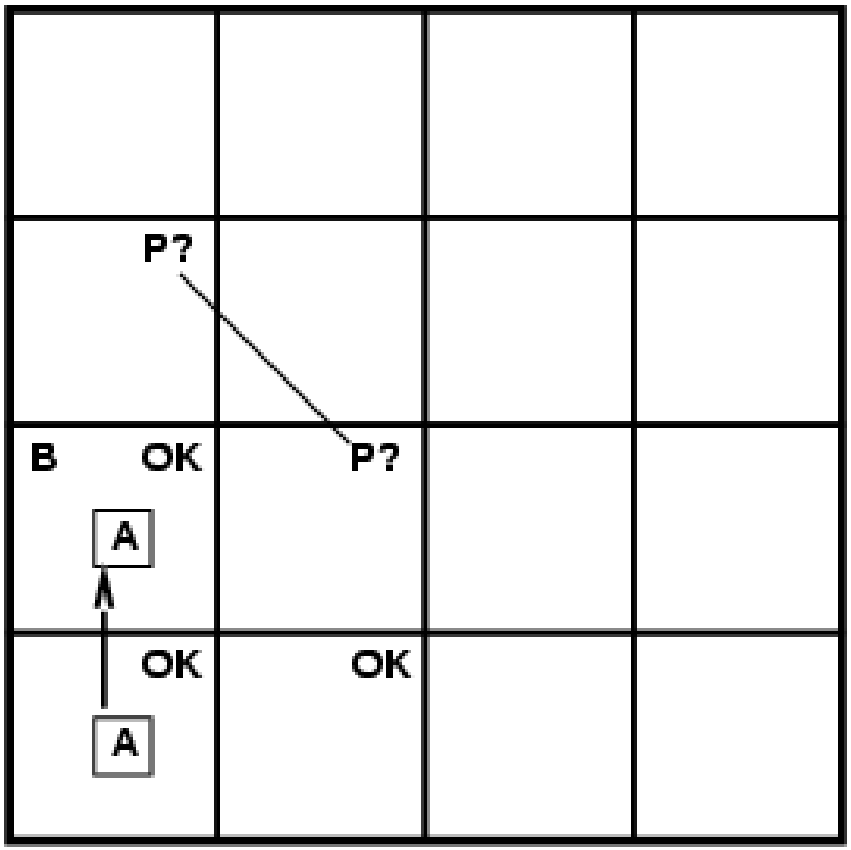
LEGEND

A agent
OK safe
P pit
W Wumpus
B breeze

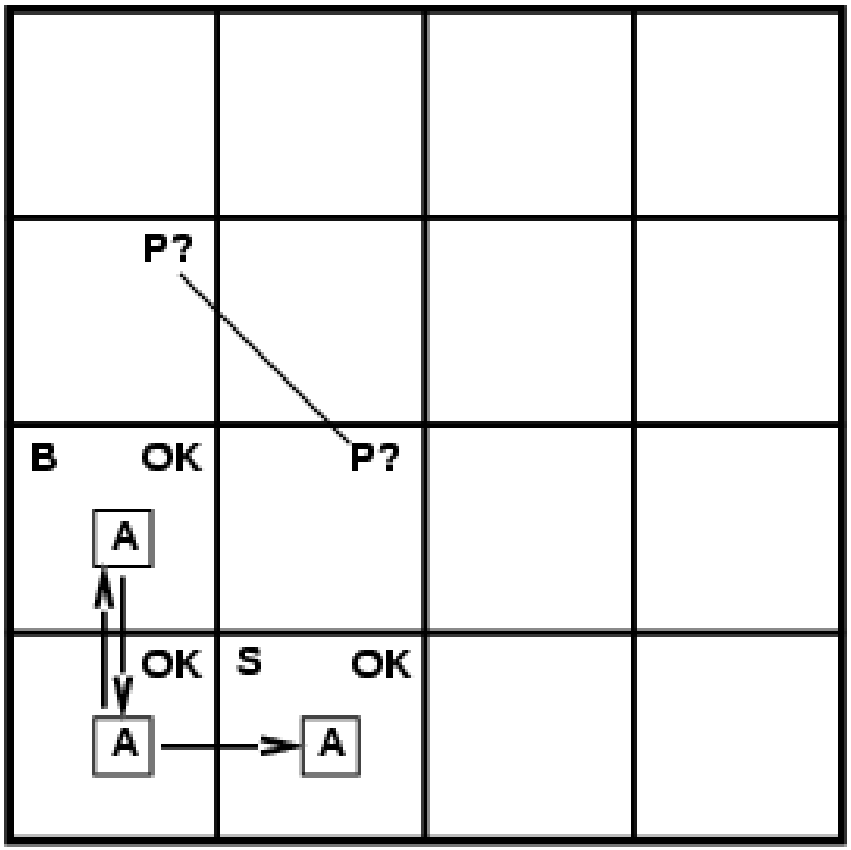
Exploring a wumpus world



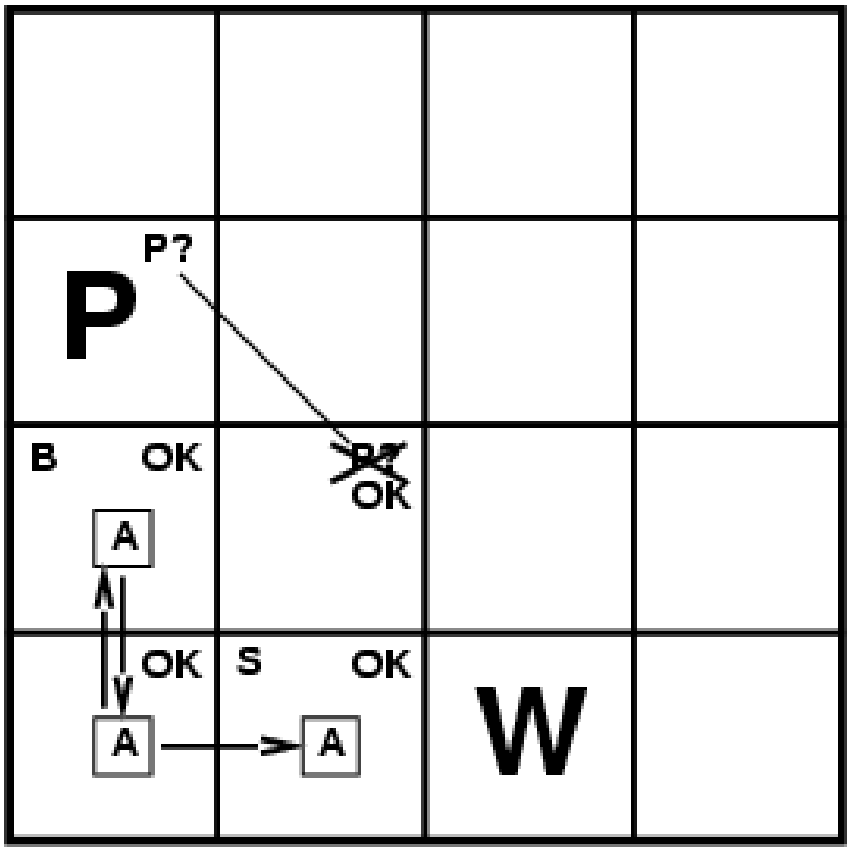
Exploring a wumpus world



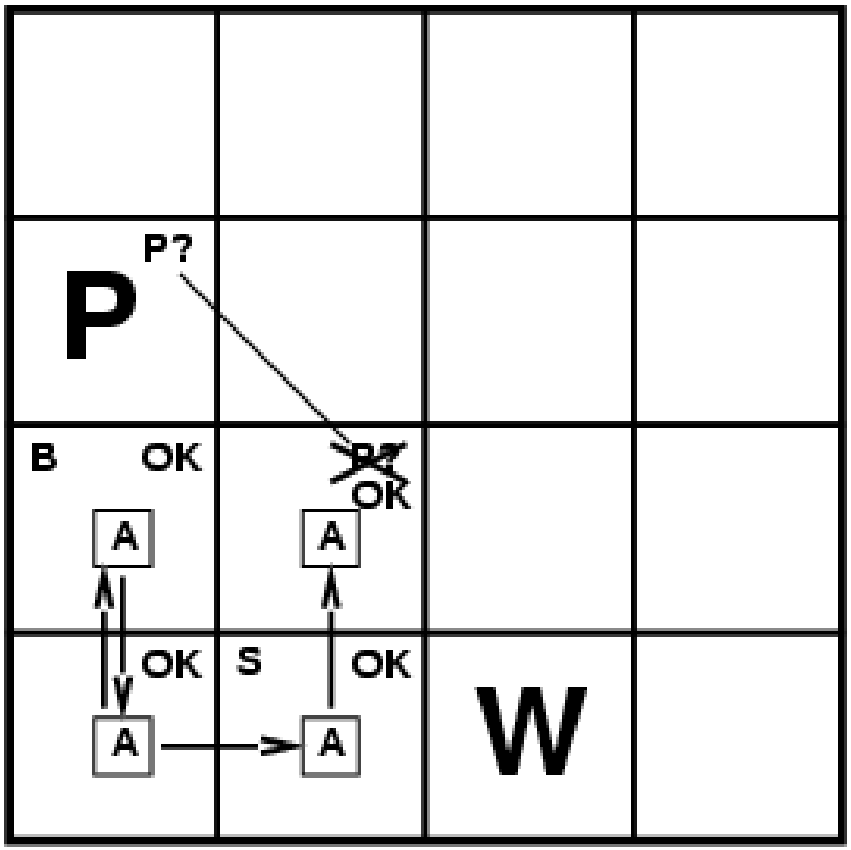
Exploring a wumpus world



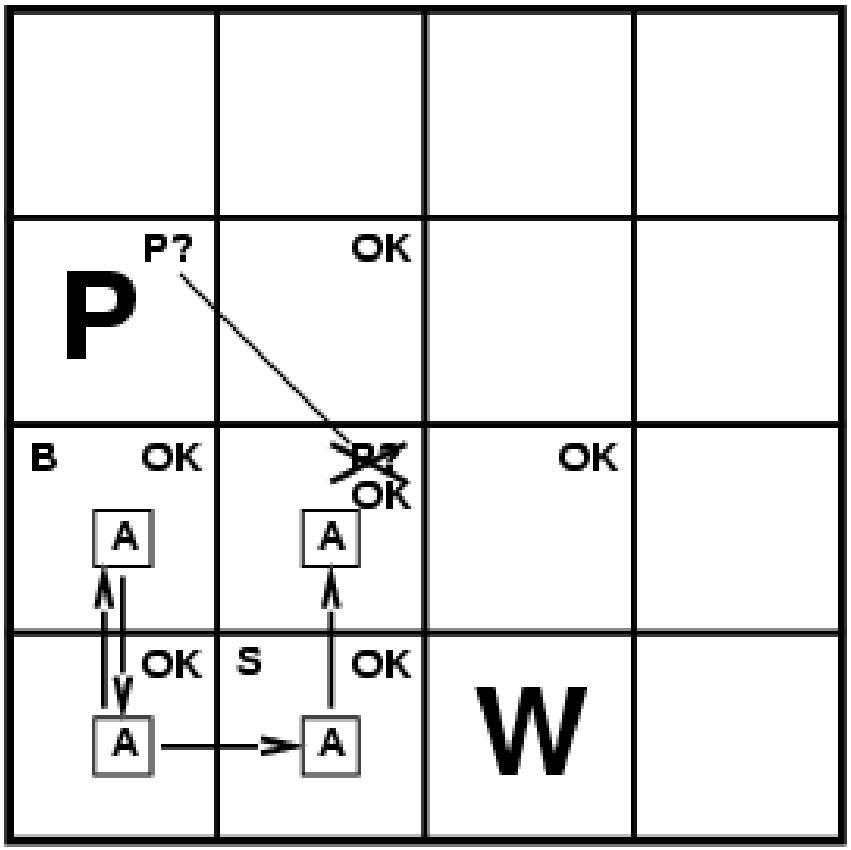
Exploring a wumpus world



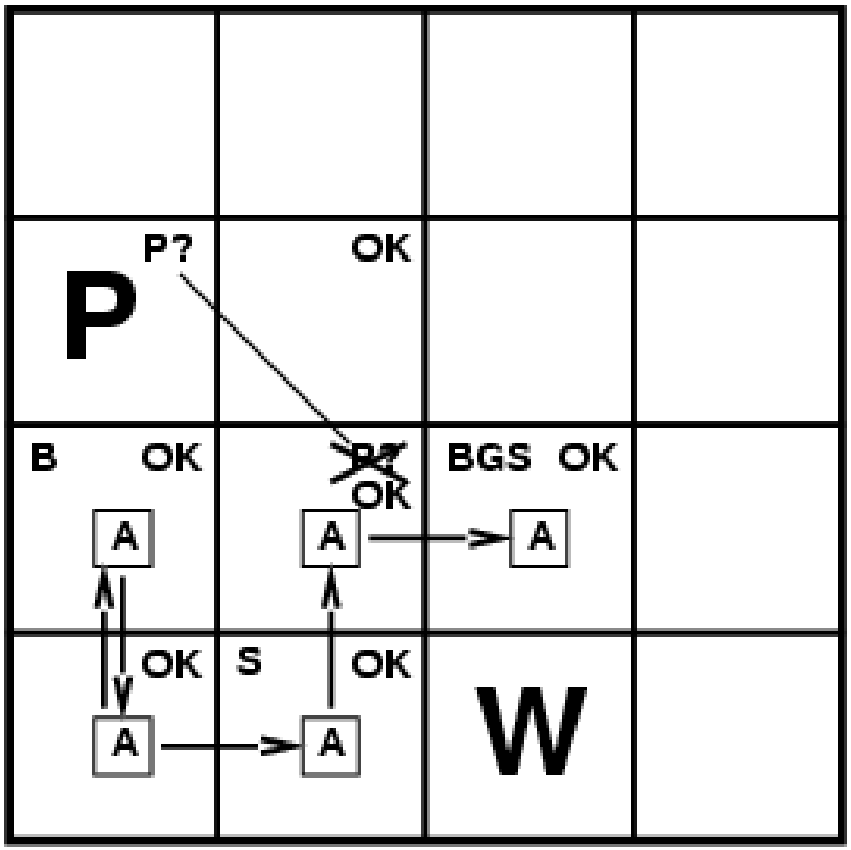
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Logic in general

Logics are formal languages for representing information such that conclusions can be drawn

Syntax defines the sentences in the language

Semantics define the "meaning" of sentences;

- i.e., define truth of a sentence in a world

E.g. for the language of arithmetic

- $x+2 \geq y$ is a sentence; $x2+y >$ is not a sentence
- $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
- $x+2 \geq y$ is true in a world where $x = 7, y = 1$
- $x+2 \geq y$ is false in a world where $x = 0, y = 6$

Entailment

Entailment means that one thing follows from another:

$KB \models \alpha$

Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

- E.g., the KB containing "Dublin won" and "Mayo won" entails "Either Dublin won or Mayo won"
- E.g., x+y = 4 entails 4 = x+y

Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

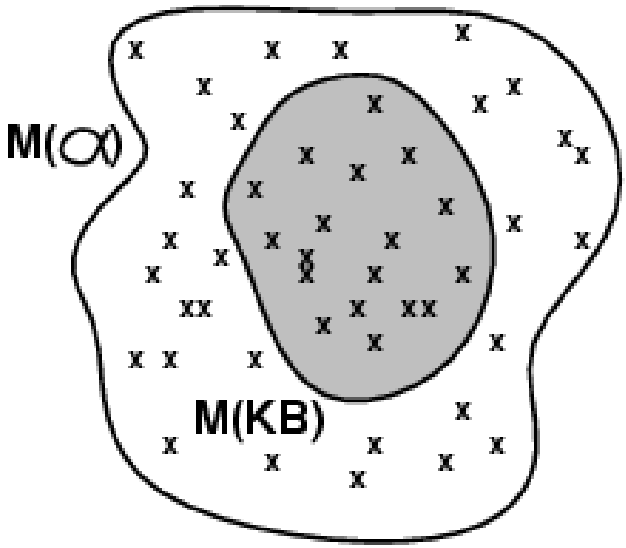
Models

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

We say *m* is a model of a sentence α if α is true in *m*

M(α) is the set of all models of α

Then KB $\models \alpha$ iff *M(KB)* $\subseteq$ *M*(a)



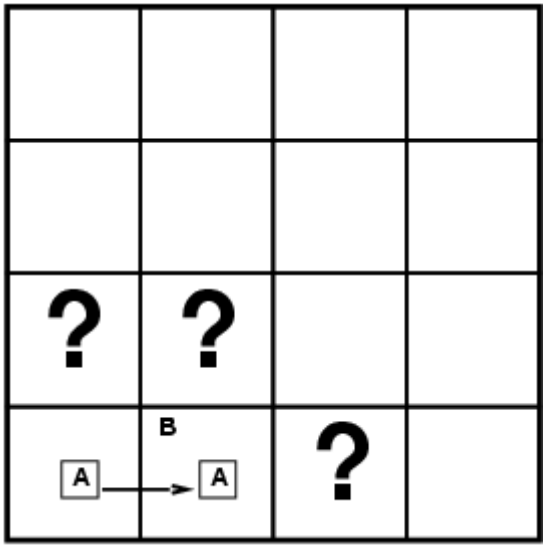
E.g.

KB = Dublin won and Mayo won

α = Dublin won

Entailment in the wumpus world

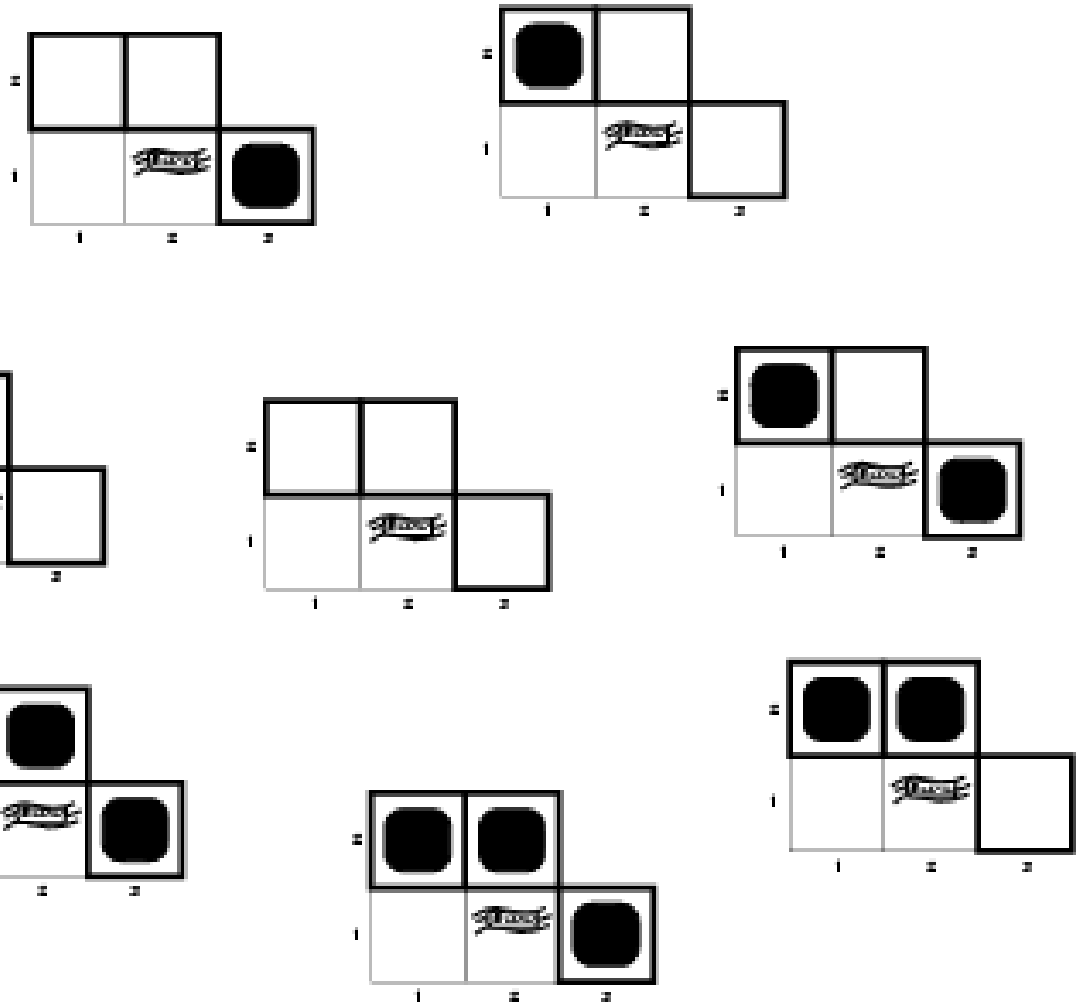
Situation after detecting nothing in [1,1],
moving right, breeze in [2,1]



Consider possible models
for *KB* assuming only pits

3 Boolean choices ⇒ 8 possible models

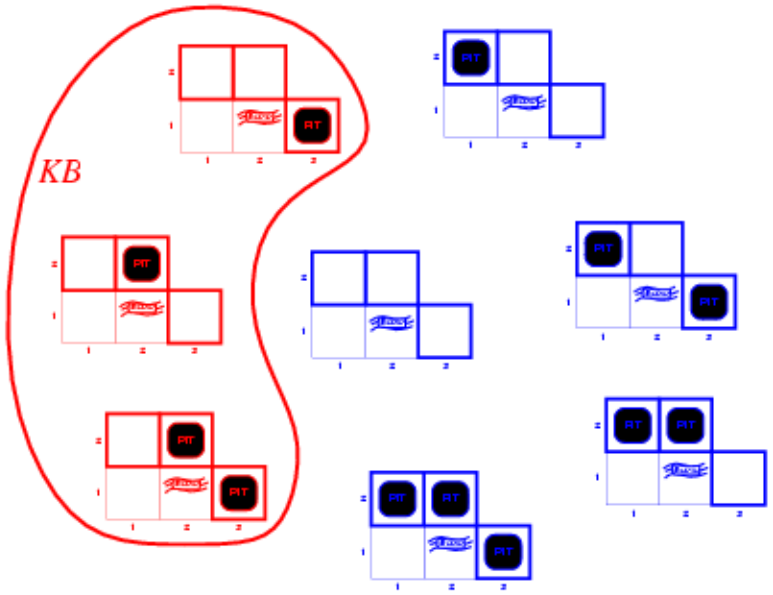
Wumpus models



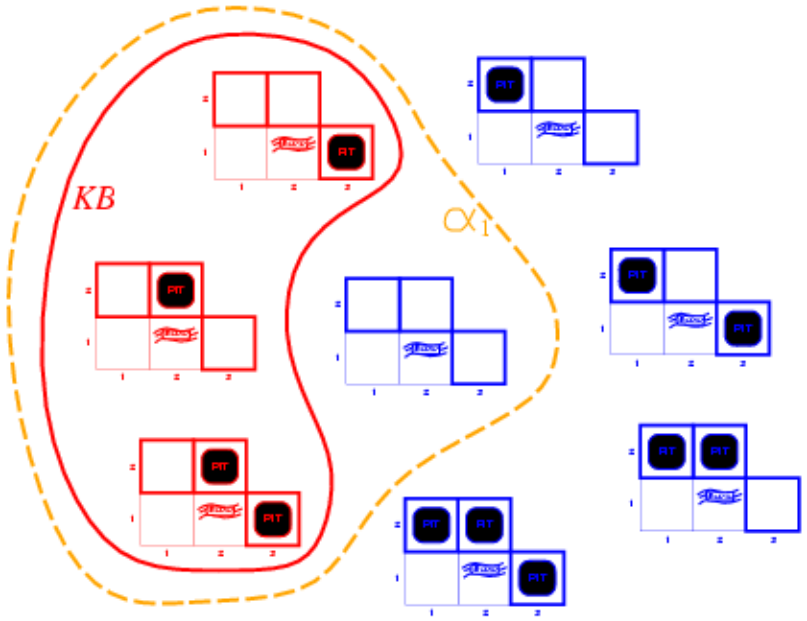
Wumpus models

A knowledge base KB is a set of propositions that the agent is given as being true

KB = wumpus-world rules +
observations



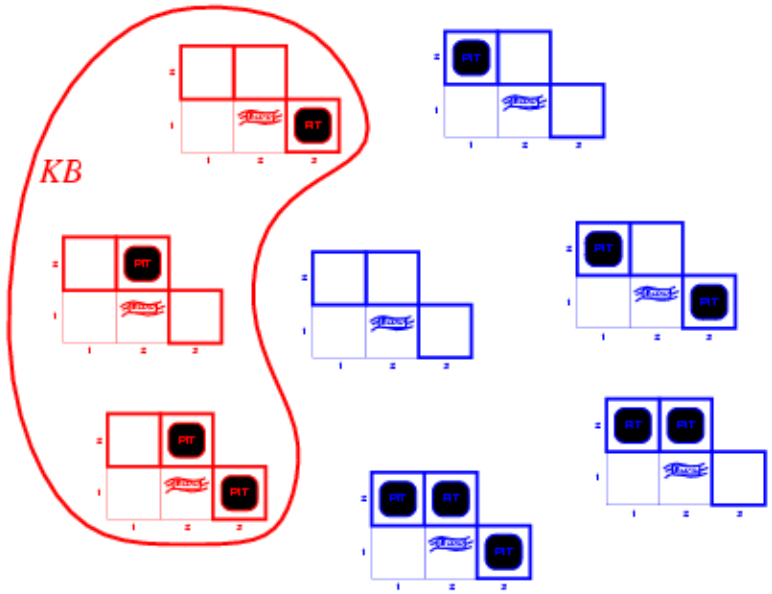
Wumpus models



KB = wumpus-world rules + observations

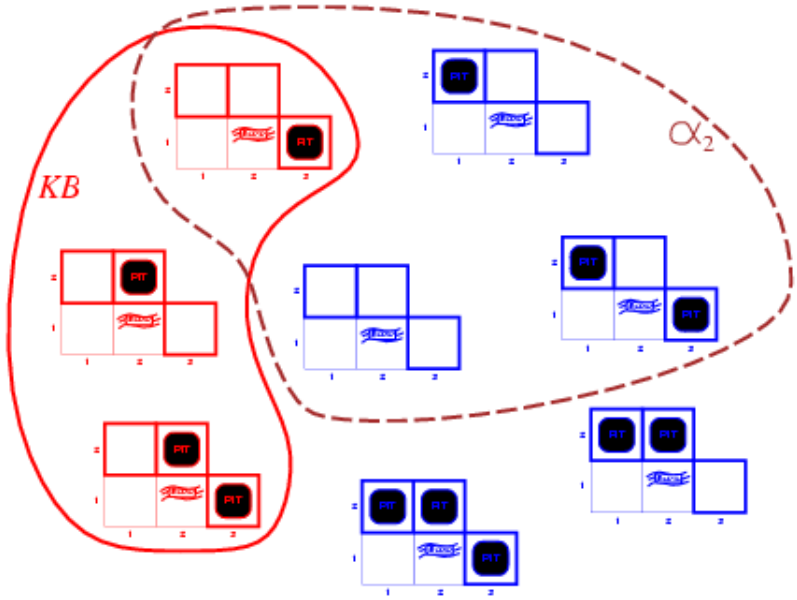
α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, proved by model checking

Wumpus models



KB = wumpus-world rules + observations

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", *KB* $\not\models \alpha_2$

Inference

A sentence is valid if and only if it is true under all possible interpretations in all possible worlds

For example,

"There is a stench in [1,1] or there is no stench in [1,1]" is valid.

A synonym for valid sentence is tautology.

Inference

A sentence is satisfiable if and only if there is some interpretation in some world for which it is true.

For example, "There is a wumpus in [1,2]" is satisfiable because there could be a wumpus there in another world, even though we don't have a wumpus there in our picture

A sentence that is not satisfiable is unsatisfiable.

For example, "There is a gold in [2,2] and there is no gold in [2,2]"

Inference

An inference algorithm is a procedure for deriving a sentence from the KB

$KB \vdash_i \alpha$

sentence α can be derived from *KB* by procedure *i*

Inference

Soundness: *i* is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

(it derives only sentences that are entailed by KB)

Completeness: *i* is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

(if it can derive any sentence that is entailed by KB)

Logic

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

Propositional logic: Syntax

Propositional logic is the simplest logic – illustrates basic ideas

1) Logical constants *True* and *False* are sentences
2) The proposition symbols P, Q, etc. are sentences
3) Wrapping bracket around a sentence yields a sentence, e.g. (P∧Q)

4) A sentence can be formed by combining simpler sentences with one of the five logical connectives
   - ∧ (and), ∨ (or), ¬ (not), ⇒ (implies) and ⇔ (bi-conditional)

Propositional logic: Syntax

A sentence can be formed by combining simpler sentences with one of the five logical connectives

- If S is a sentence, $\neg S$ is a sentence (negation)
- If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
- If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
- If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
- If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional logic: Syntax

The order of precedence (highest to lowest):

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

What is the following sentence equivalent too, if you are to put brackets?

$\neg P \vee Q \wedge R \Rightarrow S$

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	false	true	false

With these symbols 8 possible models can be enumerated automatically.

Propositional logic: Semantics

Rules for evaluating truth with respect to a model *m*:

$\neg S$ is true iff S is false
$S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true
$S_1 \vee S_2$ is true iff S_1 is true or S_2 is true
$S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true
i.e., is false iff S_1 is true and S_2 is false
$S_1 \Leftrightarrow S_2$ is true iff $S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence, e.g.

$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$ is *true* $\wedge$ (*true* $\vee$ *false*) is *true* $\wedge$ *true* is *true*

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Validity and Inference

Truth tables can be used to test for valid sentences (remember that valid sentences should be true under all possible interpretations)

For example, is the following sentence valid?

$((P \vee H) \wedge \neg H) \Rightarrow P$

P	H	P∨H	(P∨H)∧ ¬H)	((P∨H)∧ ¬H) ⇒ P
False	False	False	False	True
False	True	True	False	True
True	False	True	True	True
True	True	True	False	True

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in [i, j].
Let $B_{i,j}$ be true if there is a breeze in [i, j]

For our wumpus world we have the following sentences:

R1: $\neg P_{1,1}$

"Pits cause breezes in adjacent squares"
R2: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
R3: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

(The above are true in ANY wumpus world)

R4: $\neg B_{1,1}$
R5: $B_{2,1}$

Truth tables for inference

We have 7 propositional symbols, so we can enumerate all possible worlds, which are $2^7 = 128$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
$false$	$false$	$false$	$false$	$false$	$false$	$false$	$false$
$false$	$false$	$false$	$false$	$false$	$false$	$true$	$false$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$false$	$true$	$false$	$false$	$false$	$false$	$false$	$false$
$false$	$true$	$false$	$false$	$false$	$false$	$true$	$true$
$false$	$true$	$false$	$false$	$false$	$true$	$false$	$true$
$false$	$true$	$false$	$false$	$false$	$true$	$true$	$true$
$false$	$true$	$false$	$false$	$true$	$false$	$false$	$false$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$true$	$true$	$true$	$true$	$true$	$true$	$true$	$false$

Inference by enumeration

Let α = A ∨ B and KB = (A ∨ C) ∧ (B ∨ ¬C)

Is it the case that KB $\models \alpha$?

Check all possible models – α must be true wherever KB is true

A	B	C	KB = (A ∨ C) ∧ (B ∨ ¬C)	α = A ∨ B
True	True	True		
True	True	False		
True	False	True		
True	False	False		
False	True	True		
False	True	False		
False	False	True		
False	False	False		

Inference by enumeration

Inference by enumeration of all models is sound and complete

For n symbols, time complexity is $O(2^n)$, space complexity is $O(n)$

Logical equivalence

Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Validity and satisfiability (again!)

A sentence is valid if it is true in all models,

e.g. *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model, e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models, e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

Proof methods

Proof methods divide into (roughly) two kinds:

Application of inference rules

- Legitimate (sound) generation of new sentences from old
- Proof = a sequence of inference rule applications
  Can use inference rules as operators in a standard search algorithm
- Typically require transformation of sentences into a normal form

Model checking

- truth table enumeration (always exponential in *n*)
- improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
- heuristic search in model space (sound but incomplete)
  e.g., min-conflicts-like hill-climbing algorithms

Conjunctive Normal Form

Approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF) is a conjunction of disjunctions of literals

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Theorem: For every formula, there exists an equivalent formula in CNF

(We will see shortly how we can convert to CNFs!)

Inference: Resolution

Resolution inference rule (for CNF): $\dfrac{p \vee q,\ \neg p}{q}$

Or in general form:

$$\frac{\ell_1 \vee \ldots \vee \ell_k,\ \ m_1 \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

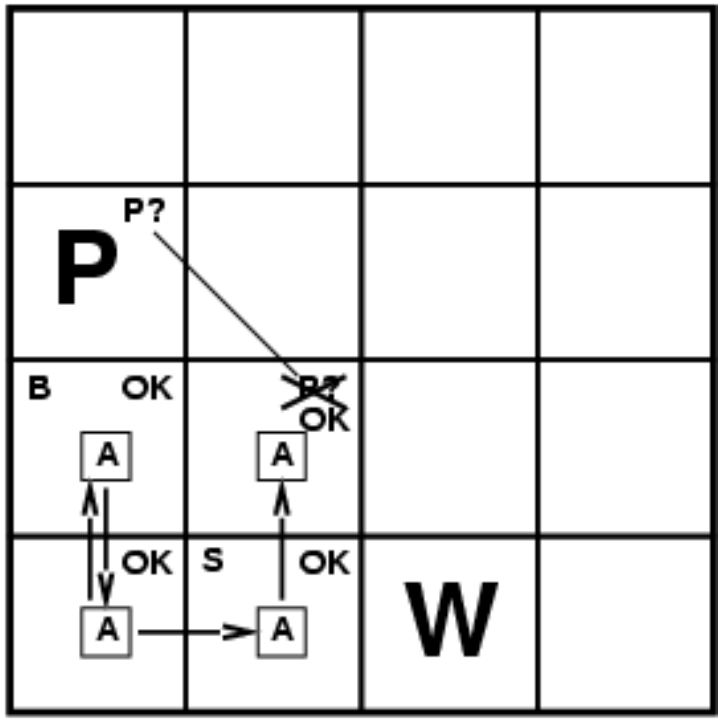
where ℓ_i and m_j are complementary literals.

Resolution is sound and complete for propositional logic

Resolution

For example

$$\frac{P_{1,3} \vee P_{2,2},\ \neg P_{2,2}}{P_{1,3}}$$



Conversion to CNF

Steps for conversion to CNF:

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.
3. Move $\neg$ inwards using de Morgan's rules and double-negation:
4. Apply distributivity law ($\wedge$ over $\vee$) and flatten

Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributivity law ($\wedge$ over $\vee$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution algorithm

Proof by contradiction: we assert the negation of what we are trying to prove, e.g. $\neg\alpha$ and try to show that $KB \wedge \neg\alpha$ unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*

clauses ← the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
new ← { }
loop do
  for each C_i, C_j **in** *clauses* **do**
    resolvents ← PL-RESOLVE(C_i, C_j)
    if *resolvents* contains the empty clause **then return** *true*
    new ← *new* ∪ *resolvents*
  if *new* ⊆ *clauses* **then return** *false*
  clauses ← *clauses* ∪ *new*

Proof by contradiction using resolution

$\boldsymbol{KB} = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1}$

$\boldsymbol{\alpha} = \neg P_{1,2}$

$KB \wedge \neg \alpha$

$(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \wedge \neg(\neg P_{1,2})$

$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}) \wedge \neg B_{1,1} \wedge P_{1,2}$

$(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1}) \wedge \neg B_{1,1} \wedge P_{1,2}$

...

...

KB in restricted form

If the sentences in the KB are restricted to some special forms some of the sound inference rules may become complete

Example: Horn form (Horn normal form)

$(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$

[Note that the above is the same as: $(B \Rightarrow A) \wedge ((A \wedge C) \Rightarrow D)$]

Two inference rules that are sound and complete with respect to propositional symbols for KBs in the Horn normal form:

- Resolution (positive unit resolution)
- Modus ponens

KB in Horn form

Horn form

a clause (disjunction of literals) with at most one positive literal

$(\neg A \vee \neg C \vee D)$

Not all sentences in propositional logic can be converted into the Horn form

The above can be re-written as $(A \wedge C) \Rightarrow D$

Modus ponens

Modus ponens:

$$\frac{B \Rightarrow A,\ B}{A}$$

More general version of the rule

$$\frac{B_1, \ldots, B_n,\quad B_1 \wedge \ldots \wedge B_n \Rightarrow A}{A}$$

Modus ponens is sound and complete with respect to propositional symbols for the KBs in the Horn normal form

- Note: no negation of a propositional symbol is allowed

Exercises

Exercise 1: Compute the truth table of $(F \vee G) \wedge \neg(F \wedge G)$

Exercise 2: Use the truth tables method to determine whether $(p \rightarrow q) \vee (p \rightarrow \neg q)$ is valid.

Exercise 3: Use the truth tables method to determine whether $(\neg p \vee q) \wedge (q \rightarrow \neg r \wedge \neg p) \wedge (p \vee r)$ (denoted with φ) is satisfiable.

Exercise 4: Reduce to Conjunctive Normal Form (CNF) the formula $\neg(\neg p \vee q) \vee (r \rightarrow \neg s)$

Exercise 5: Reduce to CNF the formula $(\neg p \rightarrow q) \rightarrow (q \rightarrow \neg r)$